

Smashing iOS Apps for Fun and Profit

Dark Luo & Sung-ting Tsai

{dark,tt}@chroot.org



About US



CHROOT



Dark Luo

Community

CHROOT - member

PHATE - Founder

Zuso Security - member

Speech

HITCON 2012 - iOS Hacking

HITCON 2009 - Game Hacking & Game
Protection Bypassing (Hackshield)

ZCamp 2008 - Reverse engineering

Interests

Reverse engineering & programming

RAT & Anti-Virus & Anti-Anti-Virus & Game
cheat engine



Sung-ting Tsai (TT)

Trend Micro

Leader of an advanced threat research team.

Core Tech Department

Speech

Black Hat USA 2011 / 2012

Codegate 2012

Syscan 10'

HITCon 08'

Research

New security technology

Malicious document

Malware auto-analyzing system (sandbox technologies)

Malware detection

System vulnerability and protection

Mobile security



Agenda

iOS Hacking



For Profit



For Fun





0x01:

Hacking iOS Apps

Analyzing Tools
Debug Tricks

iOS Apps

Mach-O

Mach-O, short for Mach object file format, is a file format for executables, object code, shared libraries, dynamically-loaded code, and core dumps.

iOS and Mac OS X .

Assembly

Arm V6 – Thumb

Arm V7 – Thumb v2

Tools

A Jail-broken iPhone

The 1st step.

GDB

Dynamic analysis

IDA Pro

Static analysis

otool

Object file (mach-o) analysis

class-dump

Generate header file from binaries



Decrypting iOS Apps (before analyzing)



Locate the encrypted section

```
# otool -l filepath | grep 'crypt'
```



Using **gdb** to launch the app



Dump the section



Write back to the app



Now you may use IDA pro to analyze the app

Decrypting iOS Apps

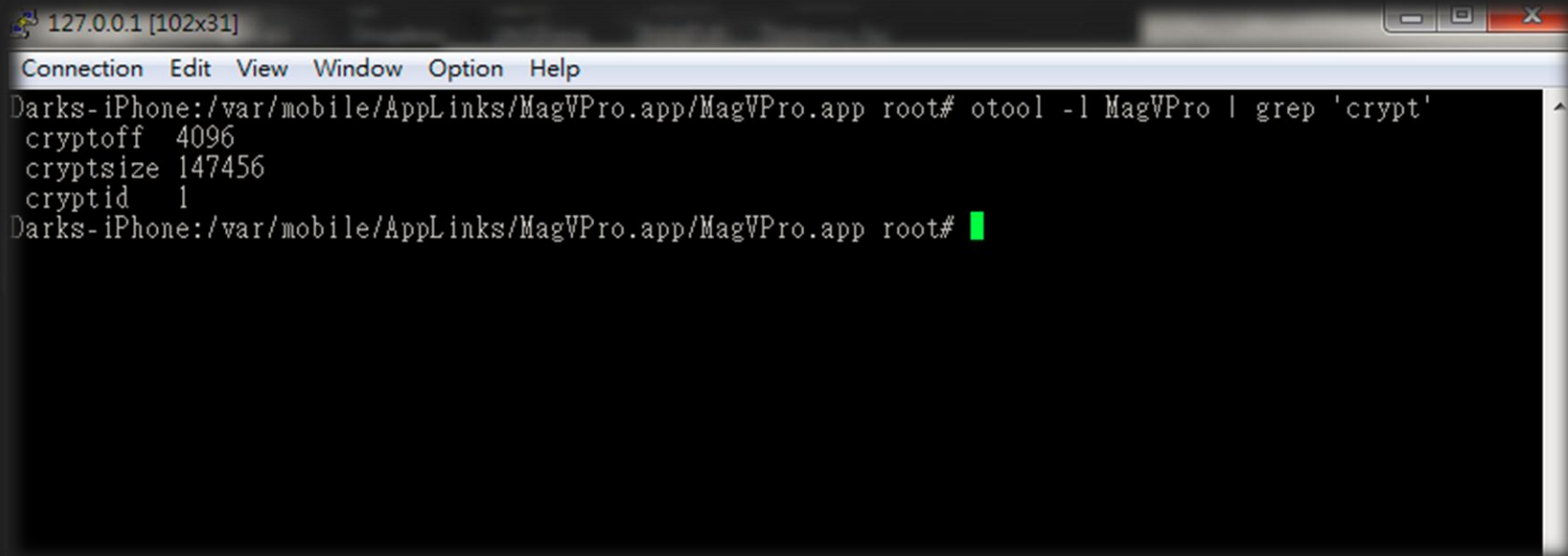


Decrypting iOS Apps (1)

```
text:00002574 ;  
text:00002574 ; =====  
text:00002574  
text:00002574 ; Segment type: Pure code  
text:00002574 AREA __text, CODE, READWRITE  
text:00002574 ; ORG 0x2574  
text:00002574 CODE32  
text:00002574  
text:00002574 EXPORT start  
text:00002574 start  
text:00002574 LDCGE p9, c12, [R6], #0x14  
text:00002578 BLLE 0xFF746890  
text:0000257C SBCNE R3, R12, R0, LSL R2  
text:00002580 LDHIB R7!, {R1-R7, R11, LR}^  
text:00002584 STMUSIB R2, {R0-R5, R7, R8, R10, R12, LR, PC}^  
text:00002588 MOUPL R11, 0xFF3BFFFF  
text:0000258C SUCCE 0xC92D71  
text:00002590 SUCGT 0xF47BF1  
text:00002594 ORRNES R6, R10, #0x2A  
text:00002598 STCLTL p0, c14, [SP, #0x26C]  
text:0000259C LDRB LR, [PC, R4, ASR#26]!  
text:000025A0 BLPL 0xFE51DD5C  
text:000025A0 ; CODE XREF: -[&Vm_____$_1____L____5____0____9k____K____1_5____: W  
text:000025A4 BCC 0xFE71BC94  
text:000025A8 LDRHIBT R4, [R2], R1, LSL#12  
text:000025AC UNAALEQS R5, R3, R10, R10  
text:000025B0 MCRHI p9, 2, R4, c14, c9, 2  
text:000025B4 STCUCL p6, c9, [R1], #-0x298
```

0001574 00002574: text: start

Decrypting iOS Apps (2)



A terminal window titled "127.0.0.1 [102x31]" with a menu bar containing "Connection", "Edit", "View", "Window", "Option", and "Help". The terminal shows a root shell on a device named "Darks-iPhone". The user runs the command `otool -l MagVPro | grep 'crypt'`, which outputs the following information:

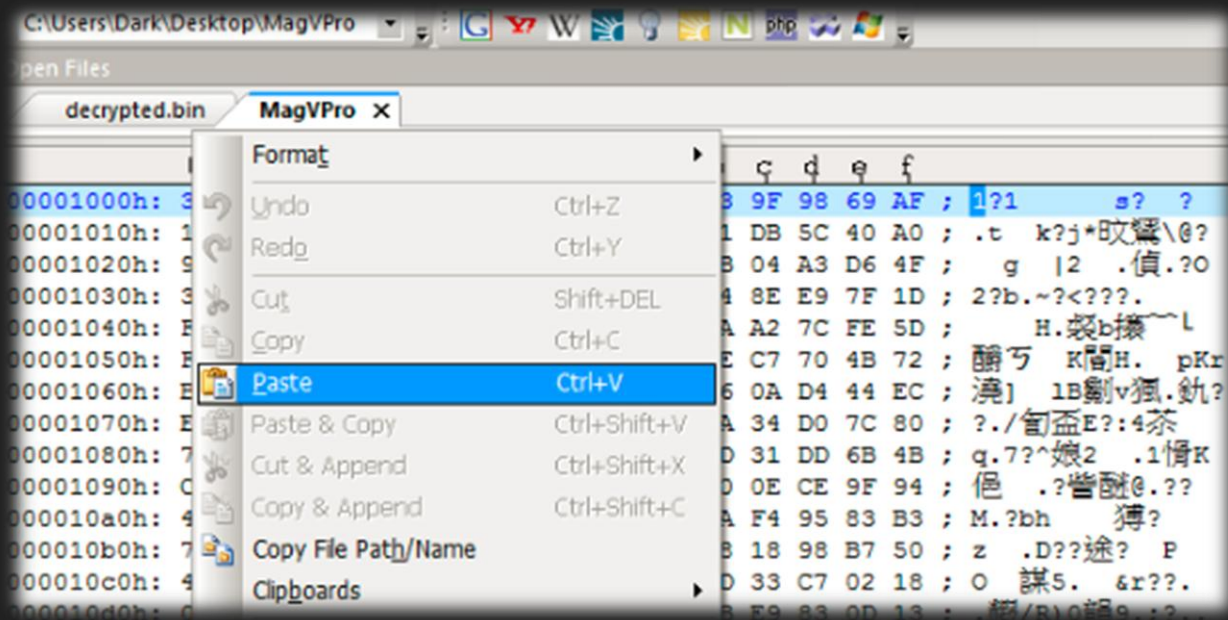
```
cryptoff 4096
cryptsize 147456
cryptid 1
```

The prompt returns to `Darks-iPhone:/var/mobile/AppLinks/MagVPro.app/MagVPro.app root#` with a green cursor.

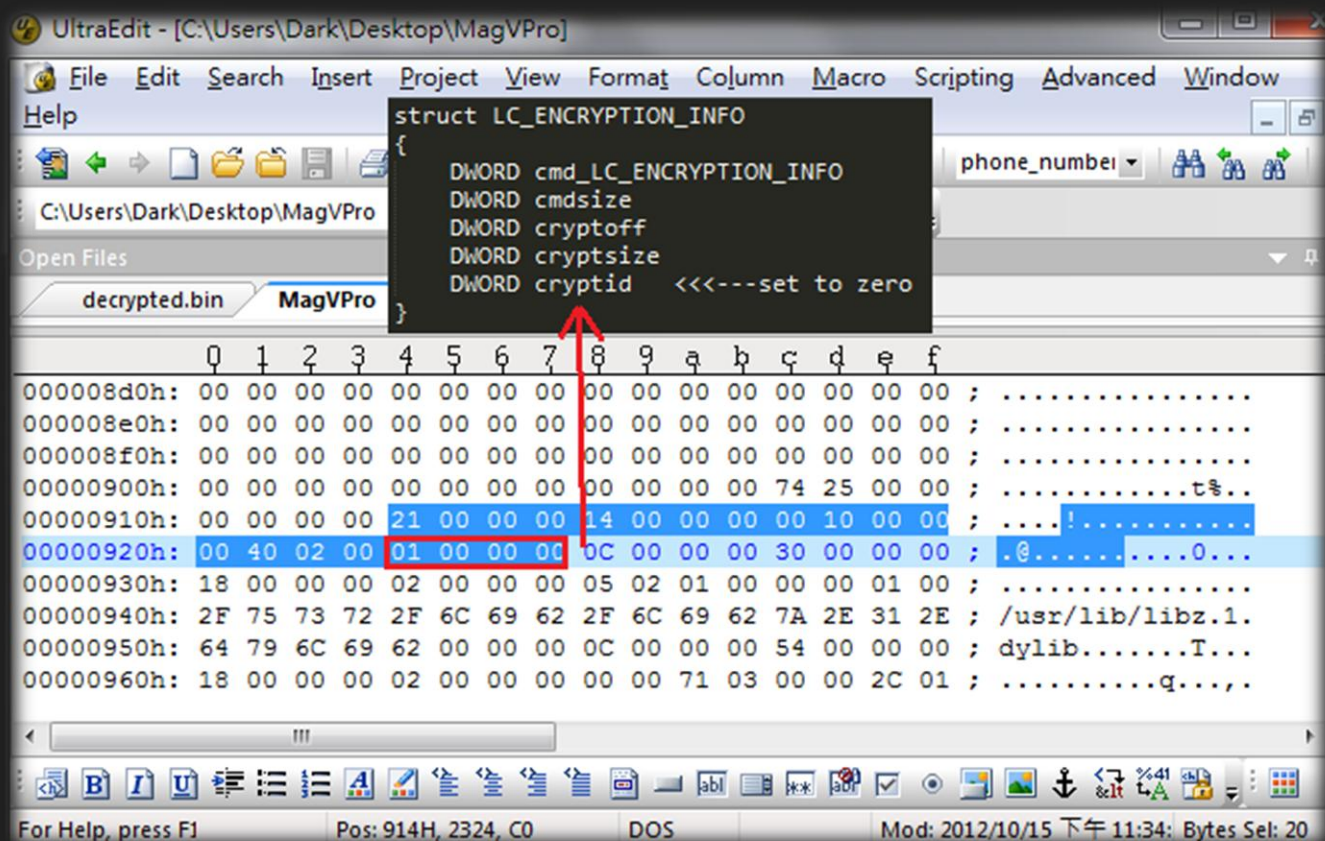
Decrypting iOS Apps (3)

```
(gdb) run
Starting program: /private/var/mobile/Applications/99008E31-BDAD-402F-AB29-CFB5A74EF544/MagVPro.app/Ma
gVPro
Removing symbols for unused shared libraries . done
Reading symbols for shared libraries .....+++.....
..... done
^C
Program received signal SIGINT, Interrupt.
0x315a9004 in mach_msg_trap ()
(gdb) dump memory decrypted.bin 0x2000 0x2000+147456
(gdb) █
```

Decrypting iOS Apps (4)

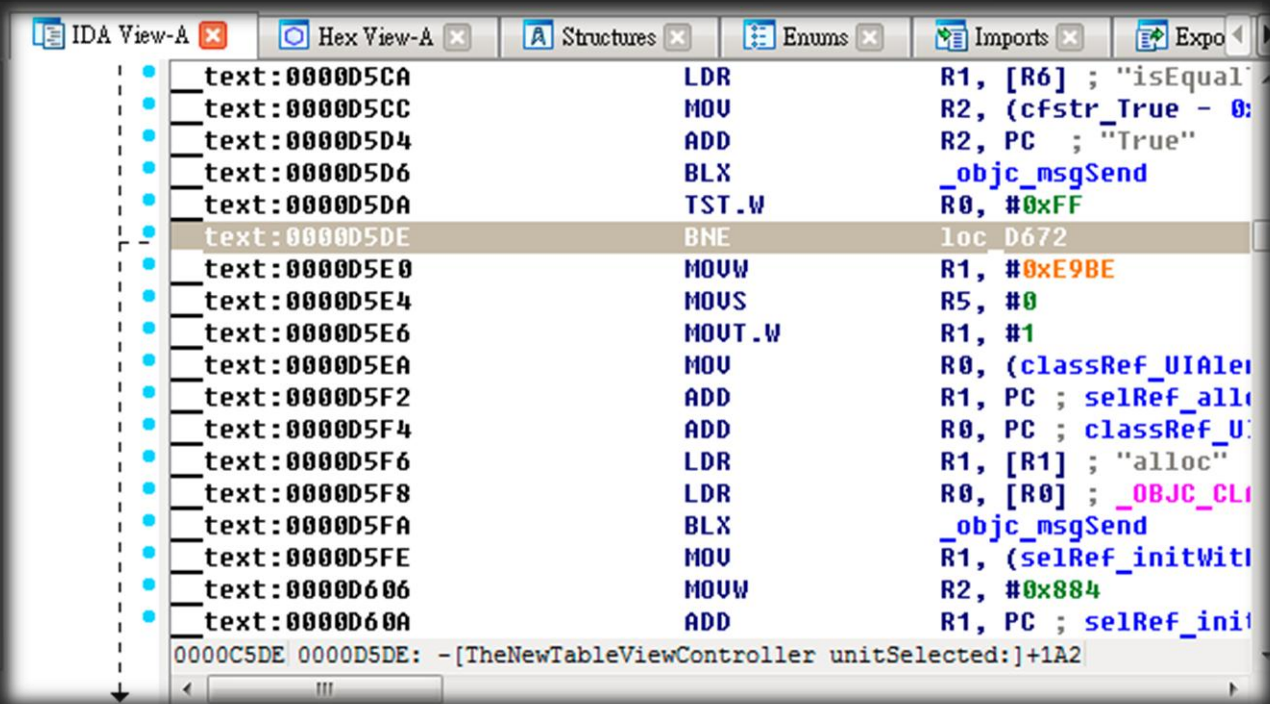


Decrypting iOS Apps (5)



Decrypting iOS Apps (6)

Done



The screenshot shows the IDA Pro interface with the 'IDA View-A' window active. The assembly code is as follows:

Address	Disassembly
text:0000D5CA	LDR R1, [R6] ; "isEqual"
text:0000D5CC	MOV R2, (cfstr_True - 0)
text:0000D5D4	ADD R2, PC ; "True"
text:0000D5D6	BLX _objc_msgSend
text:0000D5DA	TST.W R0, #0xFF
text:0000D5DE	BNE loc_D672
text:0000D5E0	MOVW R1, #0xE9BE
text:0000D5E4	MOVS R5, #0
text:0000D5E6	MOVW R1, #1
text:0000D5EA	MOV R0, (classRef_UIAler
text:0000D5F2	ADD R1, PC ; selRef_all
text:0000D5F4	ADD R0, PC ; classRef_U
text:0000D5F6	LDR R1, [R1] ; "alloc"
text:0000D5F8	LDR R0, [R0] ; _OBJC_CL
text:0000D5FA	BLX _objc_msgSend
text:0000D5FE	MOV R1, (selRef_initWitl
text:0000D606	MOVW R2, #0x884
text:0000D60A	ADD R1, PC ; selRef_ini

At the bottom, a comment line reads: 0000C5DE 0000D5DE: -[TheNewTableViewController unitSelected:]+1A2

Smashing iOS Apps

Memory
Patching

Binary Patching

IAP
Bypass

0x02:

The Profit Part

Memory Patching

Binary Patching

IAP Bypass

最新
上架



旅遊
飲食



投資
理財



Please meet the victim :)

Memory Patching

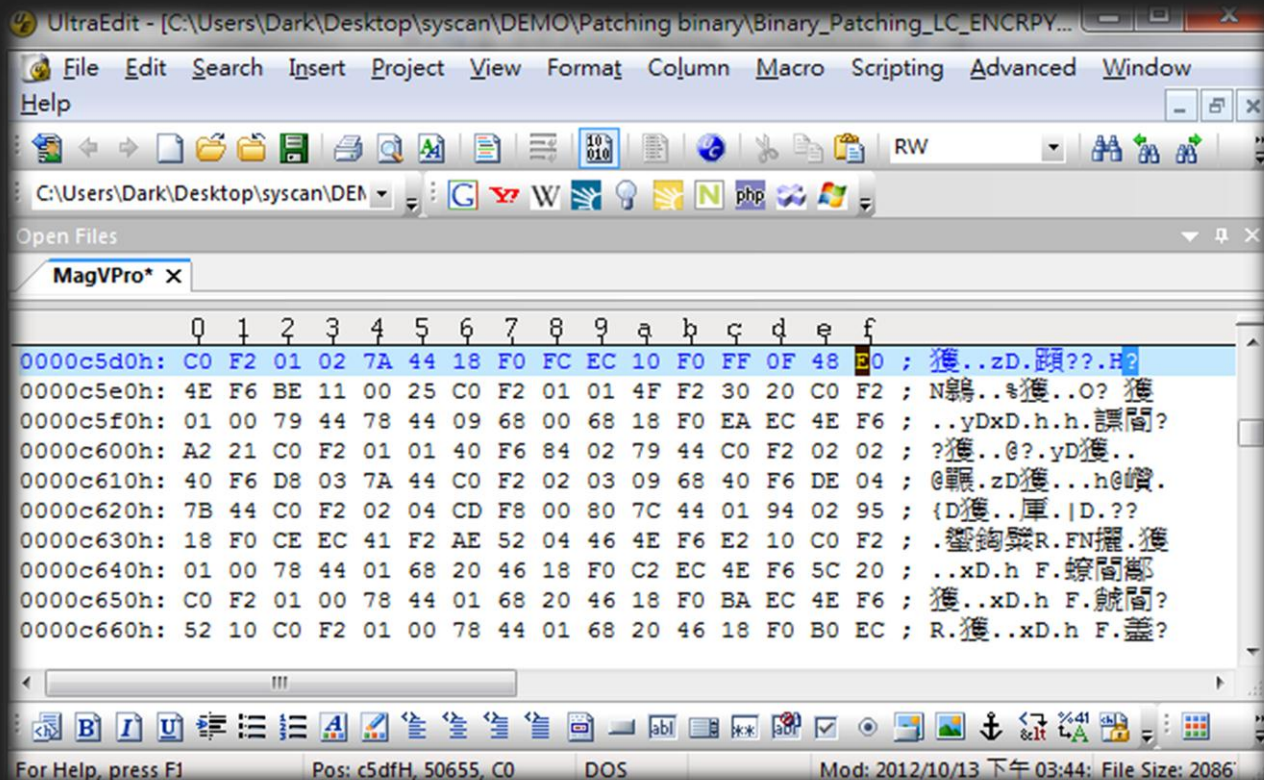
(Demo)

Binary Patching (1)

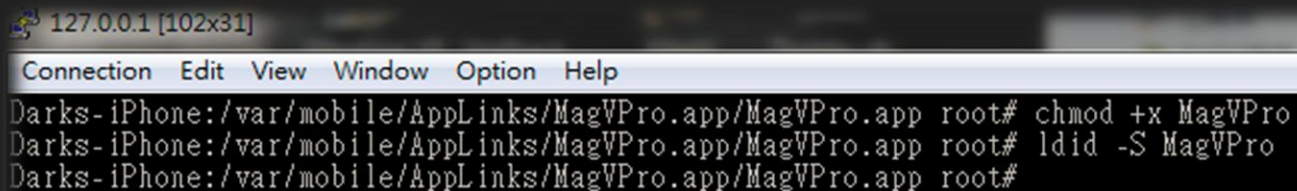
```
C:\Users\Dark\Desktop\syscan\DEMO\Patching binary\ARM_Ass...
File Edit Selection Find View Goto Tools Project Preferences Help

test.s x log.txt x
1 ARM GAS test.s page 1
2
3
4 1 0000 00000000 .org 0x000D5DE
5 1 00000000
6 1 00000000
7 1 00000000
8 1 00000000
9 2 d5de 48E0 B loc D672
10 3 d5e0 00000000 .org 0xD672
11 3 00000000
12 3 00000000
13 3 00000000
14 3 00000000
```


Binary Patching (2)



Binary Patching (3)



A screenshot of a terminal window titled "127.0.0.1 [102x31]". The window has a menu bar with "Connection", "Edit", "View", "Window", "Option", and "Help". The terminal shows three lines of commands executed as root on an iPhone:

```
Darks- iPhone:/var/mobile/AppLinks/MagVPro.app/MagVPro.app root# chmod +x MagVPro
Darks- iPhone:/var/mobile/AppLinks/MagVPro.app/MagVPro.app root# ldid -S MagVPro
Darks- iPhone:/var/mobile/AppLinks/MagVPro.app/MagVPro.app root#
```

Binary Patching (4)

Done



IAP Bypass (In-App-Purchase)

MITM SSL Proxy

No need to Jailbreak.

Doesn't work since iOS 6.

Patching IAP API

Require Jailbreak.

transactionState

SKPaymentTransactionStatePurchasing
SKPaymentTransactionStatePurchased
SKPaymentTransactionStateFailed
SKPaymentTransactionStateRestored

IAP Checking Flow

```
(void)paymentQueue:(SKPaymentQueue *)queue updatedTransactions:(NSArray *)transactions
{
    for (SKPaymentTransaction *transaction in transactions)
    {
        switch (transaction.transactionState)
        {
            case SKPaymentTransactionStatePurchased:
                if([self putStringToItunes:transaction.transactionReceipt]){
                    // many developers will stop here. (no further receipt verification)
                }
                break;
        }
    }
}
```

IAP Bypass

(Demo)



0x03:

The Fun Part

iOS RAT

Super Gamer

iOS backdoor and
Remote Administration Tool
(Demo)



RAT Implementation

Unix socket programming basically.

Turn off screen causes TCP disconnected.

Keep sending heartbeat packet to server.

Persistent

launchctl load /System/Library/LaunchDaemons/xxx.plist



Super Gamer

(Demo)



Conclusion

iOS Hacking



Techniques and
Tricks

For Profit



How bad it
could be

For Fun



Demonstrated
the possibilities



Thank you!

{dark,tt}@chroot.org